软件测试

教学做一体化教程

朱毅 王淑华 陈恒 王雅轩◎编著

清华大学出版社

软件测试

教学做一体化教程

朱毅 王淑华 陈恒 王雅轩◎编著

清华大学出版社
北京

内 容 简 介

本书采用“教、学、做”一体化的方式撰写，合理地组织学习单元，并将每个单元分解为核心知识、能力目标、任务驱动、实践环节4个模块。全书共8章，第1章是软件测试基本概念，第2章是软件测试基本技术和测试过程，第3、4章是测试方法中的黑盒测试和白盒测试，第5章是测试管理以及自动化测试，第6～8章是软件测试工具QTP、LoadRunner、JUnit的理论与实践。书中实例侧重实用性和启发性、趣味性强、分布合理、通俗易懂，使读者能够快速掌握软件测试的基础知识、测试管理技术以及软件测试工具的使用技巧，为适应实战应用打下坚实的基础。

本书适合作为高等院校“软件工程”课程的教材或教学参考书，也适合作为有一定经验的软件工作人员的参考用书。

图书在版编目(CIP)数据

软件测试教学做一体化教程/朱毅等编著. --北京：清华大学出版社，2014
ISBN 978-7-302-36593-8

Ⅰ. ①软… Ⅱ. ①朱… Ⅲ. ①软件－测试－高等学校－教材 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2014)第112112号

责任编辑：田在儒
封面设计：王跃宇
责任校对：李　梅
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795764

印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：11.5　　**字　　数**：260千字
版　　次：2014年7月第1版　　**印　　次**：2014年7月第1次印刷
印　　数：1～2500
定　　价：29.00元

产品编号：045664-01

前言
FOREWORD

本书按照“教、学、做”一体化模式精编了软件测试的核心内容，以核心知识、能力目标、任务驱动和实践环节为单元组织本书的体系结构。核心知识体现最重要和实用的知识，是教师需要重点讲解的内容；能力目标提出学习核心知识后应具备的能力；任务驱动给出了教师和学生共同来完成的任务；实践环节给出了需要学生独立完成的实践活动。

全书共8章。第1章概括地介绍了软件测试的基本概念，包括软件测试产生的背景、软件测试基础理论以及目前软件测试的岗位需求。第2章介绍了软件测试基本技术和测试过程，包括软件测试的分类方法、测试过程中的模型、软件测试阶段和开发阶段的对照关系、测试工作流程以及测试计划和测试用例的编写。第3章介绍了黑盒测试的基本原理以及主要的黑盒测试方法，包括等价类划分法、边界值分析法、决策表分析法，并结合案例进行方法讲解以及测试用例的设计。第4章介绍了白盒测试的基本原理以及主要的白盒测试方法，包括逻辑覆盖法、独立路径测试法、面向对象的白盒测试，并结合案例进行方法讲解以及测试用例的设计。第5章介绍了软件测试管理及自动化测试的基本概念，包括软件测试管理计划、软件测试管理过程、软件缺陷管理的定义以及缺陷的属性、软件测试管理周期以及自动化测试发展的必然性。第6章介绍了功能测试工具QTP的基本工作原理以及实际应用，包括QTP工作流程、检查点设置、参数化测试脚本、输出值类型等。第7章介绍了负载测试工具LoadRunner的基本工作原理以及实际应用，包括核心组件的使用、制订测试计划、创建测试脚本、定义场景、执行场景和结果分析等。第8章介绍了单元测试工具JUnit的基本工作原理以及实际应用，包括JUnit测试框架中的核心类、使用断言方法来实现单元测试等。

本书特别注重引导学生参与课堂教学活动，适合高等院校相关专业作为“教、学、做”一体化的教材。

编　者

2014年1月

目 录
CONTENTS

第1章

软件测试概述

主要内容

- 软件测试背景
- 软件测试基础理论
- 软件测试的岗位需求

在学习软件测试之前，应该具有一定的计算机编程基础、数据库管理基础以及软件工程等理论基础。学习软件测试之后，可以采用相关测试理论以及测试方法、测试工具，有针对性地对已有软件产品或者待开发的软件产品进行相关的测试工作，保证软件的质量，提高客户的满意度。在本章中主要掌握软件测试背景、软件测试的发展现状、软件测试的基础理论、软件测试的岗位需求等相关内容。

1.1 软件测试背景

1.1.1 核心知识

随着计算机与信息技术的飞速发展，软件产品已经应用到社会的各个领域之中，与人们现实中的生活息息相关，软件产品的质量自然成为人们共同关注的焦点。在软件行业激烈的竞争环境中，软件开发商为了占有市场，避免被淘汰，必须重视软件产品的质量。用户为了保证自己业务能顺利完成，也希望能够选用优质的软件。质量不佳的软件产品不仅会使开发商的维护费用和用户的使用成本大幅增加，还可能产生其他的责任风险，造成公司信誉下降。在一些关键应用（如银行系统、证券交易系统、铁路调度系统、民航订票系统、军事防御和核电站安全控制系统等）中使用质量有问题的软件，还可能造成灾难性的后果。

软件危机曾经是软件界甚至整个计算机界最热门的话题。为了解决这场危机，软件从业人员、专家和学者做出了大量的努力。现在人们已经逐步认识到所谓的软件危机实际上仅是一种状况，那就是软件中有错误，正是这些错误导致了软件开发在成本、进度和质量上的失控。软件中存在错误是不可避免的，因为软件是由人来完成的，所有由人做的工作都不会是完美无缺的。问题在于软件开发人员如何去避免错误的产生和消除已经产生的错误，使程序中的错误密度达到尽可能低的程度。

软件测试作为软件产品质量保障的重要手段之一，越来越受到软件开发商的重视，为了

保证软件质量，所以软件产品必须要进行软件测试。软件测试已经成为软件开发中必不可少的环节。统计表明，在典型的软件开发项目中，软件测试工作量往往占软件开发总工作量的40%以上，而在软件开发的总成本中，用在测试上的开销要占30%～50%。如果把维护阶段也考虑在内，讨论整个软件生存期时，测试的成本比例也许会有所降低，但实际上维护工作相当于二次开发，乃至多次开发，其中必定还包含有许多测试工作，软件开发在资金上的平均投入如图1.1所示。

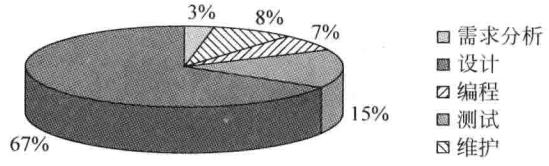

图1.1　软件开发在资金上的平均投入

从图中可以看出，测试和维护的成本之和占到软件开发总成本的80%左右。因此，测试对于软件产品来说是必需的，问题在于应该考虑采用什么方法、如何进行软件测试。

1. 软件

在测试之前，首先了解一下什么是软件。

直观上说，当人们提起软件时，通常想到的是软件产品。如Word、个人理财管家、腾讯QQ等，用户知道如何操作软件就可以。稍微有点计算机基础知识的用户，能更深一步地知道，软件是通过特定的编程语言和开发工具，组合在一起形成的程序或软件包。这种程序或者软件包通常称之为“代码”。这样的理解也多少有些片面。

权威机构IEEE(电气电子工程师学会，世界上最大的专业技术组织之一)给出了软件的定义：软件是由计算机程序、规程以及可能的相关文档和运行计算机系统所需要的数据组成。即，软件＝程序＋规程＋文档＋数据。

程序：也称为“代码”。用来激活计算机执行所需的应用。

规程：用来确定执行程序的顺序和进度安排，应用的方法。

文件：针对开发人员、用户和维护人员有不同类型的文档。开发文档(需求分析报告、设计报告、程序注释文档等)使得开发组成员能够进行高效的合作，能够对软件设计与最终产品进行有效的评审和审查。用户文档(用户手册)使用户知道如何使用软件产品。维护文档(程序员软件手册)向维护组提供每个软件模块的结构和任务所需要的信息，在软件产品出现错误(Bug)时，能快速地定位错误起因以及对软件进行及时更新和维护。

数据：包括代码、参数、数据表等数据。另一种重要数据是标准测试数据，用来测试软件的功能、性能是否存在一定的问题。

2. 软件缺陷及原因

软件缺陷(Bug)是指计算机系统或者程序中存在的任何一种破坏其正常运行能力的问题、错误或者隐藏的功能缺陷、瑕疵。

一个Bug影响力到底有多大呢？下面的事实也许更具有说服力。

(1)“爱国者”导弹防御系统。美国爱国者导弹防御系统1991年首次被用在第一次海湾战争对抗伊拉克“飞毛腿”导弹的防御作战中，总体上看效果不错，赢得各界的赞誉，但它

还是有几次失利，没有成功拦截伊拉克“飞毛腿”导弹，其中一枚在沙特阿拉伯的多哈爆炸的“飞毛腿”导弹造成 28 名美国士兵死亡。分析专家发现，拦截失败的症结在于一个软件缺陷，当“爱国者”导弹防御系统的时钟累计运行超过 14 小时后，系统的跟踪系统就不准确了。在多哈袭击战中，“爱国者”导弹防御系统运行时间已经累计超过 100 多个小时，显然那时的跟踪系统已经很不准确，从而造成这种结果。

(2) 英特尔奔腾芯片缺陷。在计算机的“计算器”程序中输入以下算式：(4195835/3145727)×3145727－4195835，如果答案是 0，就说明该计算机浮点运算没问题；如果答案不是 0，就表示计算机的浮点除法存在缺陷。1994 年，英特尔奔腾 CPU 芯片就曾经存在这样一个软件缺陷，而且被大批生产出来卖到用户那里。最后，英特尔为自己处理软件缺陷的行为道歉，并拿出 4 亿多美元来支付更换坏芯片的费用。

(3) “冲击波”计算机病毒。“冲击波”计算机病毒首先在美国发作，使美国的政府机关、企业及个人用户的成千上万计算机受到攻击。随后，冲击波蠕虫很快在因特网上广泛传播，中国、日本和欧洲等国家和地区也相继受到不断的攻击，结果造成十几万台邮件服务器瘫痪，给整个世界范围内的 Internet 通信带来惨重损失。后来调查发现，制造冲击波蠕虫的黑客仅仅用了 3 周时间就制造了这个恶毒的程序，“冲击波”计算机病毒仅仅是利用微软 Messenger Service 中的一个缺陷，即可以攻破计算机安全屏障，可使基于 Windows 操作系统的计算机崩溃。该缺陷几乎影响当前所有微软 Windows 系统，它甚至使安全专家产生更大的忧虑：独立的黑客们将很快找到利用该缺陷控制大部分计算机的方法。随后，微软公司不得不紧急发布补丁包，修正这个缺陷。

美国商务部下属的国力标准技术研究所(NIST)有关软件缺陷的调查报告显示：“由于软件缺陷引起的损失额每年高达 595 亿美元，这一数字相当于美国国民生产总值的 0.6%。”

软件缺陷会造成软件质量的下降，最终导致软件产品在某种程度上不能满足用户的需求。软件缺陷可以是代码缺陷、规程缺陷、文件缺陷或者软件数据缺陷等。所有引起这些缺陷的原因都是人，是由系统分析员、程序员、软件测试人员、文档专家、客户经理，甚至有的时候是由客户和他们的代表造成的。

软件缺陷存在于软件产品的整个生存周期中，通常是由以下 9 个方面引起的。

(1) 需求定义不完全。软件需求定义不完全是产生软件缺陷的最主要原因之一。此类最常见的问题有：需求的错误性定义；缺少一些重要的需求；需求定义的不完备；包含了不必要的需求等。

(2) 客户与开发人员沟通不畅。客户与开发人员沟通不畅是开发过程早期阶段出现的主要问题。此类最常见的问题有：系统分析员没有针对不确定的需求和客户进行反复确认；客户对需求表述不清；客户的需求变更没有落实在书面上；客户对开发者提出的设计问题存在误解等。

(3) 开发人员对软件需求的偏离。在一些情况下，软件开发者可能偏离文档化的需求，这种行为经常引起软件缺陷。常见的情况有：开发者重用以前项目中的模块，却没有对用户新需求进行更改和进行适当的分析；由于时间或者经费问题，开发者为应对开发压力删去部分需求的功能；开发者未得到客户的批准，对软件进行改进等。

(4) 逻辑设计错误。当设计系统的专业人员(系统分析师、软件工程师、分析员等)系统

地阐述软件需求时，软件缺陷就可能进入系统。典型的错误有：通过错误的算法表达软件需求；过程前后顺序的错误；对系统边界条件定义的错误；漏掉需要的软件系统状态；漏掉定义软件系统非法操作的反应等。

(5) 编码错误。各种各样的原因可以使程序员产生编码错误。包括误解设计文档；编程语言中的语言性错误；开发工具的应用错误；数据选择错误等。

(6) 不符合文档编写规范。几乎所有软件公司都有自己的文档编制与编码规范，这些标准定义了文档的内容、次序和格式以及程序员编写的代码。为了支持这种要求，软件公司会公布文档模板和编码规范，要求所有开发组和成员必须遵守这些要求。不遵守文档规范，会使软件的后期处理增加出错的几率。

(7) 测试过程不足。测试过程的不足会留下大量未检测到和未改正的错误，从而影响出错率。这些不足主要来自于：不完备的测试计划；检测到的错误和缺陷未记入文档和报告；由于没有合适地指出缺陷原因，未能及时改正检测到的软件缺陷等。

(8) 规程错误。规程向用户指引每个处理步骤所需的活动。它们对复杂软件系统尤其重要，因为处理是分若干个步骤进行的，每一步都要输入各种类型的数据并允许检查中间结果。

(9) 文档编制错误。给开发和维护组制造麻烦的通常都是文档编制错误，在设计文档中的错误会最终体现在软件产品中。另一类文档编制错误主要影响用户，如在用户手册和软件“帮助”中显示的错误，会造成用户的不正确地使用软件产品。

软件缺陷按照对用户使用影响的严重级别可以分为以下几种。

(1) 建议型：通常为可用性方面的一些建议，如字体颜色等一些不影响使用的问题。

(2) 提示型：软件产品中存在一些小问题，如有个别错别字、文字排版不整齐等，对功能几乎没有影响，软件产品仍可使用。

(3) 一般型：软件产品中存在不太严重的错误，如次要功能模块丧失、提示信息不够准确、用户界面差和操作时间长等。

(4) 严重型：软件产品中存在严重错误，指功能模块或特性没有实现，主要功能部分丧失，次要功能全部丧失或致命的错误声明。

(5) 致命型：软件产品中存在致命的错误，造成系统崩溃、死机或造成数据丢失、主要功能完全丧失等。

3. 软件质量

软件测试的目的是保证软件产品的质量。在分析了软件缺陷的原因后，那什么是软件质量呢？

IEEE 指出：软件质量是系统、部件或过程满足规定需求的程度，是满足顾客或用户需求的程度。

软件质量本质上就是软件产品符合用户的需求，达到用户的满意度。影响软件质量的主要因素从管理角度可划分为以下三组。

(1) 从产品运行上可以分为：正确性、健壮性、效率、完整性、可用性、风险；

(2) 从产品修改上可以分为：可理解性、可维修性、灵活性、可测试性；

(3) 从产品转移上可以分为：可移植性、可再用性、互运行性。

考量软件质量主要有以下 11 个指标。

(1) 性能(Performance)是指系统的响应能力,即要经过多长时间才能对某个事件作出响应,或者在某段时间内系统所能处理的事件个数;

(2) 可用性(Availability)是指系统能够正常运行的时间比例;

(3) 可靠性(Reliability)是指系统在应用或者错误面前,在意外或者错误使用的情况下维持软件系统功能特性的能力;

(4) 健壮性(Robustness)是指在处理或者环境中系统能够承受的压力或者变更能力;

(5) 安全性(Security)是指系统向合法用户提供服务的同时能够阻止非授权用户使用的企图或者拒绝服务的能力;

(6) 可修改性(Modification)是指能够快速地以较高的性能价格比对系统进行变更的能力;

(7) 可变性(Changeability)是指体系结构扩充或者变更成为新体系结构的能力;

(8) 易用性(Usability)是衡量用户使用软件产品完成指定任务的难易程度;

(9) 可测试性(Testability)是指软件发现故障并隔离定位其故障的能力特性,以及在一定的时间或者成本前提下进行测试设计、测试执行的能力;

(10) 功能性(Function ability)是指系统能完成所期望工作的能力;

(11) 互操作性(Inter-Operation)是指系统与外界或系统与系统之间的相互作用能力。

4. 软件质量保证

软件质量保证(Software Quality Assurance,SQA)即参照一定的质量标准、目标及各项软件流程、规范来监督、管理公司软件产品的质量;它的目的是为了客观地核实软件项目的实施行动与开发中的产品是否遵从于对应的需求、过程描述、标准及规程。

软件质量保证的目标主要包括以下 4 个方面:

(1) 通过监控软件开发过程来保证产品质量;

(2) 保证开发出来的软件和软件开发过程符合相应标准与规程;

(3) 保证软件产品、软件过程中存在的不合理问题得到处理,必要时将问题反映给高级管理者;

(4) 确保项目组制订的计划、标准和规程适合项目组需要,同时满足评审和审计需要。

SQA 与两种不同的参与者相关:一种是做技术工作的软件工程师,另一种是负责质量保证的计划、监督、记录、分析及报告工作的 SQA 小组。

软件工程师通过采用可靠的技术方法和措施,进行正式的技术评审,执行计划周密的软件测试来考虑质量问题,并完成软件质量保证和质量控制活动。

SQA 小组的职责是辅助软件工程小组得到高质量的最终产品。SQA 小组主要完成以下 6 项工作。

(1) 为项目准备 SQA 计划。该计划在制定项目规定、项目计划时确定,由所有感兴趣的相关部门评审。

(2) 参与开发项目的软件过程描述。评审过程描述以保证该过程与组织政策、内部软件标准、外界标准以及项目计划的其他部分相符。

(3) 评审各项软件工程活动,对其是否符合定义好的软件过程进行核实,记录、跟踪与过程的偏差。

(4) 审计指定的软件工作产品，对其是否符合事先定义好的需求进行核实。对产品进行评审，识别、记录和跟踪出现的偏差；对是否已经改正进行核实；定期将工作结果向项目管理者报告。

(5) 确保软件工作及产品中的偏差已记录在案，并根据预定的规程进行处理。

(6) 记录所有不符合的部分并报告给高级领导者。

1.1.2 能力目标

掌握软件缺陷的定义以及产生原因；掌握软件质量的定义及考核指标；掌握软件质量保证的定义及软件质量保证的工作流程。

1.1.3 任务驱动

1. 分组讨论什么是好的软件产品。

2. 关于软件测试对软件质量的意义，有以下观点：①度量与评估软件的质量；②保证软件质量；③改进软件开发过程；④发现软件错误。你认为哪些观点是正确的？并说明原因。

1.1.4 实践环节

1. 通过网络搜索曾经发生软件缺陷的重要事件并分析其产生的原因。

2. 接触一种软件产品，写出该软件产品的主要功能以及使用感受，并分析该软件可能存在的缺陷，在哪些方面可以进行改进。

1.2 软件测试基础理论

1.2.1 核心知识

1. 软件测试的定义

软件测试是通过人工或者自动化手段来运行或测试某个系统的过程，它是验证软件是否能够达到客户期望功能的唯一有效方法，也是保证软件产品质量的唯一途径。软件测试并非是简单的“挑错”，而是贯穿于软件生产过程的始终，是一套完善的质量体系。这要求测试工程师具备系统的测试专业知识及对软件的整体把握能力。

软件测试概念的确定，曾经经历过以下两种方法的争论。

(1) 20 世纪 70 年代初期，Bill. Hetzel 提出“评价一个程序和系统的特性或能力，并确定它是否达到预期的结果。软件测试就是以此为目的的任何行为。”他的核心观点是：测试是验证软件是“工作的”，以正向思维，针对软件系统的所有功能点，逐个验证其正确性。这就是软件测试的第一类方法。

(2) Glenford. Myers 提出“测试不应着眼于验证软件是工作的，应该认定软件是有错误的，然后用逆向思维去发现尽可能多的错误。测试是为发现错误而执行的一个程序或者系统的过程。”他的核心观点是：测试是为了证明程序有错，而不是证明程序无错误；一个好的测试用例是在于它能发现至今未发现的错误；一个成功的测试是发现了至今未发现的错

误的测试。这是软件测试的第二类方法。

目前广泛采用的是第二类方法。

2. 软件测试的目的

基于不同的立场，存在着两种完全不同的测试目的。从用户的角度出发，普遍希望通过软件测试暴露出软件中隐藏的错误和缺陷，以考虑是否可以接受该产品。而从软件开发者的角度出发，则希望测试成为表明软件产品中不存在错误的过程，验证该软件已正确地实现了用户的要求，确立用户对软件质量的信心。

因为在程序中往往存在着许多预料不到的问题，可能会被疏漏，许多隐藏的错误只有在特定的环境下才可能暴露出来。如果不把着眼点放在尽可能查找错误这样一个基础上，这些隐藏的错误和缺陷就查不出来，会遗留到运行阶段中去。如果站在用户的角度替他们设想，就应当把测试活动的目标对准揭露程序中存在的错误。在选取测试用例时，主要考虑那些易于发现程序错误的数据。

下面这些观点可以看作是测试的目的或定义：

(1) 测试是为了发现程序中的错误而执行程序的过程；

(2) 好的测试方案是发现迄今为止尚未发现的错误的测试方案；

(3) 成功的测试是发现了至今为止尚未发现的错误的测试。

从上述规则可以看出，测试的正确定义是“为了发现程序中的错误而执行程序的过程”。这和某些人通常想象的“测试是为了表明程序是正确的”，“成功的测试是没有发现错误的测试”等观点是完全相反的。正确认识测试的目的是十分重要的，测试目的决定了测试方案的设计。如果为了表明程序是正确的而进行测试，就会设计一些不易暴露错误的测试方案；相反，如果测试是为了发现程序中的错误，就会力求设计出最能暴露错误的测试方案。

由于测试的目的是暴露程序中的错误，从心理学角度看，由程序的编写者自己进行测试是不恰当的。因此，在集成测试阶段通常由其他人员组成测试小组来完成测试工作。此外，应该认识到测试绝不能证明程序是正确的。即使经过了最严格的测试之后，仍然可能还有没被发现的错误潜藏在程序中。测试只能查找出程序中的错误，不能证明程序中没有错误。

3. 软件测试的原则

在软件测试的过程中，通常应该遵循以下 7 个原则。

(1) 所有的测试都应追溯到用户需求。这是因为软件的目的是使用户完成预定的任务，满足其需求；而软件测试揭示软件的缺陷和错误，一旦修正这些错误就能更好地满足用户需求。

(2) 应尽早地和不断地进行软件测试。由于软件的复杂性和抽象性，在软件生命周期各阶段都可能产生错误，所以不应把软件测试仅仅看作是软件开发的一个独立阶段，而应当把它贯穿到软件开发的各个阶段中去。在需求分析和设计阶段就应开始进行测试工作，编写相应的测试计划及测试设计文档，同时坚持在开发各阶段进行技术评审和验证，这样才能尽早发现和预防错误，杜绝某些缺陷和错误，提高软件质量。测试工作进行得越早，越有利于提高软件的质量，这是预防性测试的基本原则。

(3) 在有限的时间和资源下进行完全测试并找出软件所有的错误和缺陷是不可能的，软件测试不能无限进行下去，应适时终止。因为，测试输入量大、输出结果多、路径组合多，

用有限的资源来达到完全测试是不现实的。

(4) 测试只能证明软件存在错误而不能证明软件没有错误,测试无法显示潜在的错误和缺陷,继续进一步测试可能还会找到其他错误和缺陷。

(5) 充分关注测试中的集群现象。在测试的程序段中,若发现的错误数目比较多,则残存在该程序段中的错误数目也会比较多,因此应当花较多的时间和代价测试那些具有更多错误数目的程序模块。

(6) 程序员应避免检查自己的程序。考虑到人们的心理因素,自己揭露自己程序中的错误是件不愉快的事,自己不愿意否认自己的工作;此外,由于思维定式,自己难以发现自己的错误。因此,测试一般由独立的测试部门或第三方机构进行,这样测试相对比较客观。

(7) 尽量避免测试的随意性。软件测试是有组织、有计划、有步骤的活动,要严格按照测试计划进行,要避免测试的随意性。

4. 软件测试的经济性

人们通常认为,开发一个程序是困难的,测试一个程序则比较容易。这其实是误解。设计测试用例是一项细致并需要高度技巧的工作,稍有不慎就会顾此失彼,发生不应有的疏漏。无论采用何种测试方法,由于测试情况数量巨大,都不可能进行完备的测试。

所谓完备测试,就是让被测程序在一切可能的输入情况下全部执行一遍。通常也称这种测试为"穷举测试"。在实际测试中,穷举测试工作量太大,实践上行不通,这就注定了一切实际测试都是不彻底的。当然就不能够保证被测试程序中不存在遗留的错误。

软件测试的总目标是充分利用有限的人力和物力资源,高效率、高质量地完成测试。为了降低测试成本,选择测试用例时应注意遵守"经济性"的原则。

(1) 要根据程序的重要性和一旦发生故障将造成的经济损失来确定它的测试等级。

(2) 要认真研究测试策略,以便能使用尽可能少的测试用例,发现尽可能多的程序错误。掌握好测试量是至关重要的,一位有经验的软件开发管理人员在谈到软件测试时曾这样说过:"不充分的测试是愚蠢的,而过度的测试是一种罪孽。"测试不足意味着让用户承担隐藏错误带来的危险,过度测试则会浪费许多宝贵的资源。

测试是软件生存周期中费用消耗最大的环节。测试费用除了测试的直接消耗外,还包括其他的相关费用。测试量的多少主要受以下几个因素影响。

(1) 软件系统目标。软件系统目标的差别在很大程度上影响所需要进行的测试数量。那些可能产生严重后果的系统必须要进行更多的测试。如,一个操作系统软件的测试量应该比一个应用软件系统更大。一个安全性级别较高的系统应该比一个普通的软件系统要求有更多的测试。

(2) 潜在的用户数量。一个系统的潜在用户数量在很大程度上也会影响测试数量。如,一个支持百万人同时在线购物的系统会比一个只在办公室中运行的有几百个用户的系统需要更多的测试。如果这两个系统出现问题的话,前一个系统的经济影响肯定比后一个系统要大得多。除此以外,在处理错误的时候,所花的代价的差别也很大。如果在内部系统中发现了一个严重的错误,在处理错误的时候的费用就相对少一些;如果要处理一个遍布全世界的错误就需要花费相当大的财力和精力。

(3) 信息的价值。在考虑测试的必要性时,还需要将系统中所包含的信息的价值考虑

在内，一个支持许多家大银行或众多证券交易所的客户机/服务器系统中含有经济价值非常高的内容。很显然这一系统需要比一个支持便利店的系统要进行更多的测试。这两个系统的用户都希望得到高质量、无错误的系统，但是前一种系统的影响比后一种要大得多。因此应该从经济方面考虑，投入与其经济价值相对应的时间和金钱去进行测试。

（4）开发机构。一个没有标准和缺少经验的开发机构很可能开发出充满错误的系统。在一个建立了标准化流程和有很多经验的开发机构开发出来的系统中的错误相对会少很多，因此，对于不同的开发机构来说，所需要的测试量也是截然不同的。

（5）测试的时机。测试量会随时间的推移发生改变。在一个竞争激烈的市场里，争取时间是制胜的关键，开始可能不会在测试上花多少时间，但几年后如果市场分配格局已经建立起来了，那么产品的质量就变得更重要了，测试量就要加大。测试量应该针对合适的目标进行调整。

5. 软件测试终止的标准

软件测试的经济性决定了软件测试不可能无休止地进行下去，必须有其终止的标准。具体的测试终止标准要依据各个公司具体情况来制定，不能一概而论。通常测试终止满足以下几个原则。

（1）基于“测试阶段”的原则

每个软件的测试一般都要经过单元测试、集成测试、系统测试这几个阶段，测试人员可以分别对单元测试、集成测试和系统测试制定详细的测试结束点。每个测试阶段符合结束标准后，再进行下一个阶段的测试。

例如：单元测试，通常要求测试结束点必须满足“核心代码 100%经过 Code Review”、“功能覆盖率达到 100%”、“代码行覆盖率不低于 80%”、“不存在 A、B 类缺陷”、“所有发现缺陷至少 60%都纳入缺陷追踪系统且各级缺陷修复率达到标准”等标准。

（2）基于“测试用例”的原则

测试设计人员设计测试用例，并请项目组成员参与评审，测试用例一旦评审通过，在后期测试时，就可以作为测试结束的一个参考标准。

例如：在测试过程中，如果发现测试用例通过率太低，可以拒绝继续测试，等待开发人员修复后再继续后期的测试。在功能测试用例通过率达到 100%，非功能性测试用例达到 95%以上时，允许正常结束测试。但是使用该原则作为测试结束点时，把握好测试用例的质量非常关键。

（3）基于“缺陷收敛趋势”的原则

软件测试的生命周期中随着测试时间的推移，测试发现的缺陷图线，首先成逐渐上升趋势，然后测试到一定阶段，缺陷又呈下降趋势，直到发现的缺陷几乎为零或者很难发现缺陷为止。可以通过缺陷的趋势图线的走向，来决定测试是否可以结束，这也是一个判定标准。如图 1.2 所示，随着测试工作量的不断增加，缺陷数量呈下降趋势。

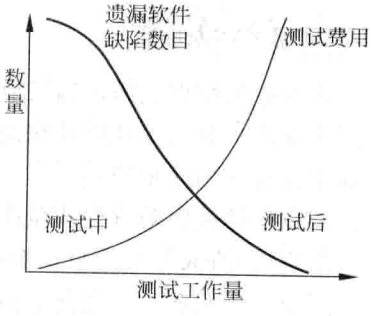

图 1.2　软件缺陷趋势图

（4）基于“缺陷修复率”的原则

软件缺陷在测试生命周期中可以分成几个严重等

级，分别是：严重错误、主要错误、次要错误、一般错误、较小错误和测试建议 6 种。测试人员在确定测试结束点时，必须保证严重错误和主要错误的缺陷修复率达到 100%，不允许存在功能性的错误；次要错误和一般错误的缺陷修复率必须达到 85%以上，允许存在少量功能缺陷，后面版本解决；对于较小错误的缺陷修复率最好达到 60%～70%以上。对于测试建议的问题，可以暂时不用修改。

(5) 基于“验收测试”的原则

当软件测试进行到一定阶段后，达到或接近测试部门指定的标准时，就可以递交用户做验收测试。如果通过用户的测试验收，就可以立即终止测试。

1.2.2 能力目标

掌握软件测试的含义和测试目的；掌握软件测试的基本原则；了解软件测试的经济性；了解软件测试终止的常见标准。

1.2.3 任务驱动

1. 简述软件测试的意义。

2. 以下说法正确的有哪些？

(1) 因为测试工作简单，对软件产品影响不大，所以可以把测试作为新员工的一个过渡工作，或安排不合格的开发人员做测试。

(2) 测试是为了证明软件的正确性。

(3) 测试过程中应重视测试的执行，可以轻视测试的设计。

(4) 测试不能修复所有的软件故障。

(5) 测试工作是在交付软件产品的时候才进行的一项活动。

1.2.4 实践环节

1. 通过到软件企业实践调研，了解软件测试工作在软件产品生存周期中的比重和作用。

2. 对软件测试的经济性进行分析，进而分析“软件测试的资金投入不用过多”观点的正确性。

1.3 软件测试的岗位需求

1.3.1 核心知识

随着中国软件业的迅猛发展，软件产品的质量控制与质量管理正逐渐成为企业生存与发展的核心。为了保证软件在出厂时的“健康状态”，几乎所有的 IT 企业在软件产品发布前都需要大量的质量控制工作。作为软件质量控制中的重要一环，软件测试工程师应运而生。国内软件业因对软件质量控制的重要作用认识较晚，尚未形成系统化的软件测试人才需求供应链，造成了目前企业欲招纳软件测试人才，却“千金难求”的尴尬局面。

判断一个职业是否有前途需要以发展的眼光分析，既要看到短期的工资待遇，更要看到未来的发展空间；既要看到短期市场需求，更要看到长远的社会需求；既要看到职业的社

会地位，更要考虑到个人的职业兴趣。软件测试行业顺应全球化和信息化发展趋势，符合我国信息化与工业化发展目标，是新兴的朝阳职业。优秀的测试从业者依靠软件测试的专业技术，可以获得职业的不断提升，随着测试能力的提升，薪资待遇不断提升，成为受人尊敬的测试专家。

1. 软件测试职业规划

每个测试从业人员都希望通过努力，提高工作职位，实现个人价值。软件测试从业者有哪些岗位可以不断提高和发展呢？软件测试网的专家将软件测试职业进行全方位分析，提出了测试职业发展具有多级别、多层次、多方向、多职位的“四多”特征。软件测试人员职业发展的路线图如图 1.3 所示。

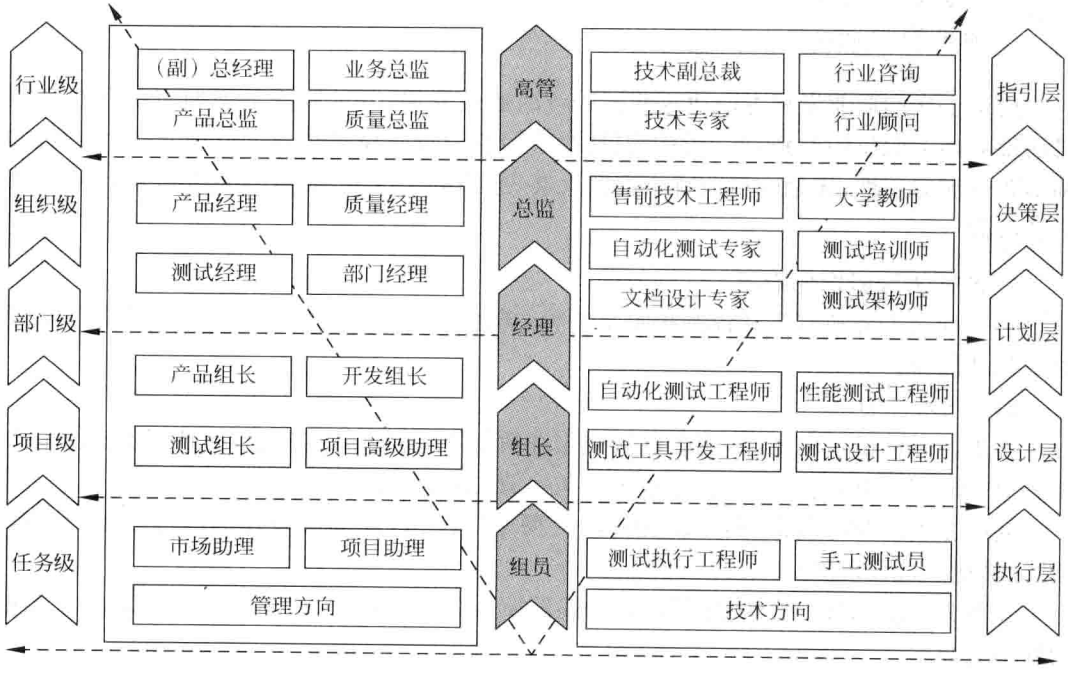

图 1.3　软件测试职业发展规划图

（1）级别角度

描述了测试工作的影响范围，从小到大的各个级别分别是“任务级”、“项目级”、“部门级”、“组织级”和“行业级”。最小的测试工作影响范围只能影响到某个具体的测试任务，最高的测试工作可以影响到测试行业的发展趋势。

（2）层次角度

描述了测试工作在组织结构中的所在地位，从低到高的各个层次分别是“执行层”、“设计层”、“计划层”、“决策层”和“指引层”。测试工作最底层是软件测试的具体执行工作，最高层是测试工作可以指引测试行业的发展。

（3）方向角度

描述了测试工作的技能发展倾向，可以分为“技术”和“管理”两个方向。“技术”方向是在测试技术、领域技术和软件工程技术的广度和深度方面进行发展。“管理”方向是向提高